Università degli Studi di Bologna
Facoltà di Ingegneria

Progettazione di Applicazioni Web T

Esercitazione 5
Il pattern DAO: un primo esercizio di base

Agenda

- **Pattern DAO**
  - **esercizio guidato** relativo alla «gestione dati degli studenti»
  - veloce ripasso di alcuni concetti chiave già visti a lezione
  - **proposta di esercizio da svolgere in autonomia** relativo alla «gestione dei corsi universitari»

Progetto di esempio

- Il file ***05_PAWeb.zip*** contiene lo scheletro di un semplice progetto di esempio basato sull'uso del pattern DAO
- Importare il progetto come visto nelle precedenti esercitazioni, senza estrarre l'archivio su file system (lo farà Eclipse)
  - *File → Import → General → Existing Projects into Workspace → Next → Select archive file*
- Progetto (nella sua versione «completa») per la gestione del database di corsi universitari
  - elenco dei **corsi attivi**
  - elenco degli **studenti**
  - **mapping tra corsi e studenti che li frequentano**
- **!** L'esercizio guidato si concentra su «studenti» e riprende la classe **StudentRepository** già vista per la metodologia «forza bruta»

Metodologia “forza bruta”

- Nella scorsa esercitazione abbiamo visto come sia possibile gestire una tabella di studenti tramite API JDBC
- Metodi Java appositamente creati per
  - connettersi a database
  - generare/eliminare tabelle
  - modificare tali tabelle (insert, update, delete)
  - interrogare tali tabelle (ad esempio query per ottenere tutti gli studenti con un certo cognome)
- Utilizzare direttamente le API JDBC è certamente possibile, ma generalmente porta a codice
  - poco leggibile
  - di difficile manutenzione
  - complesso in fase di estensione (ad esempio, per supportare un nuovo DBMS)

Il pattern DAO

- Uso di pattern (ad esempio, DAO) rende più «facile» l'accesso a DBMS
  - uso di tecniche di programmazione che rendono più semplice e meno *error-prone* l'accesso a database
  - accesso a database trasparente (per quanto possibile) rispetto al particolare DBMS in uso
  - riconfigurazione facile e veloce se si vuole utilizzare un DBMS diverso da quello precedentemente in uso

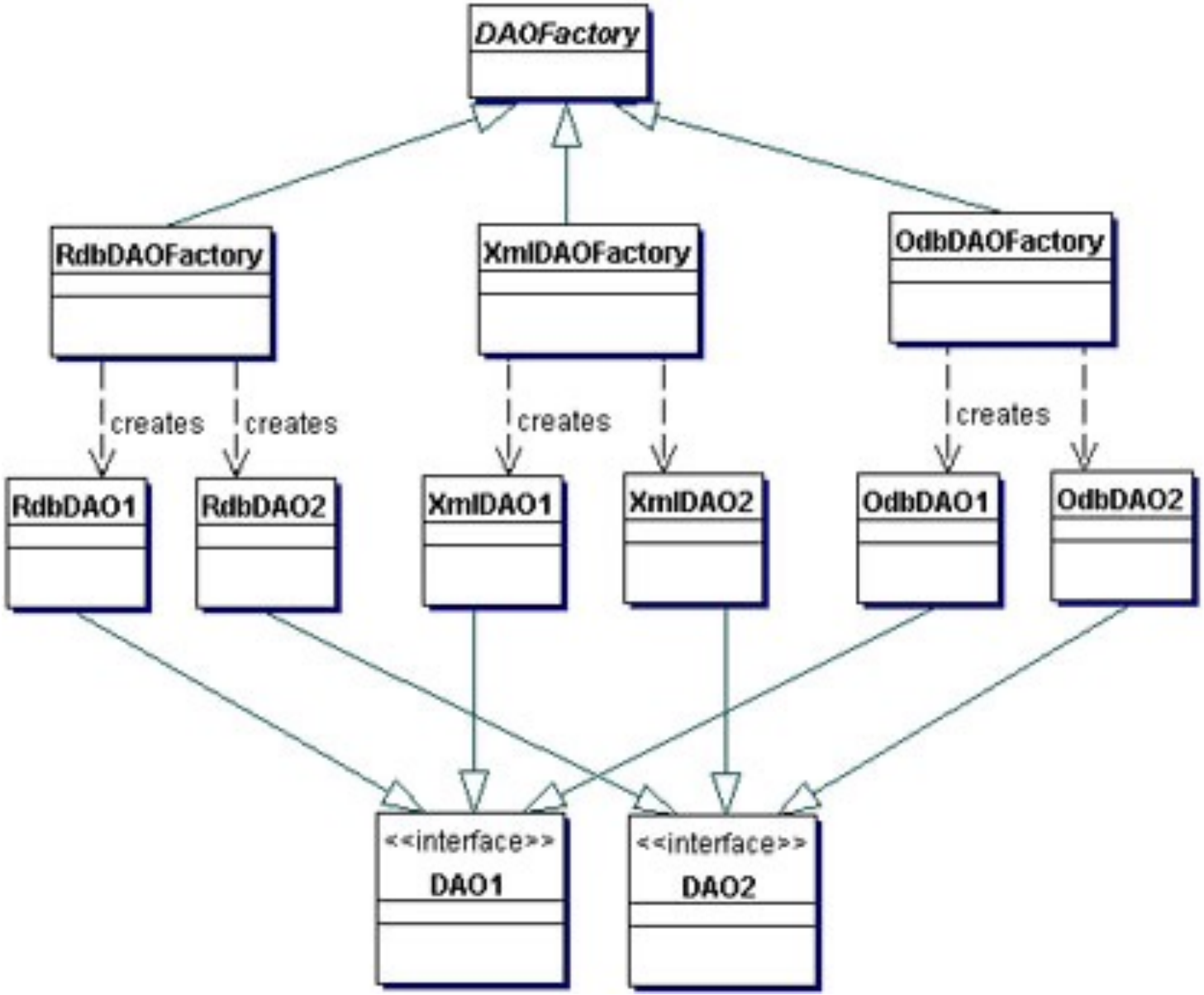
Pattern DAO con classi Factory (1)

Veloce ripasso di alcuni concetti chiave già visti a lezione:

- DTO: oggetto scambiato come Java Bean
- DAO: componente che offre i metodi per scambiare DTO tra applicazione Java e DBMS

- Una unica factory astratta
  - fornisce specifiche per le factory concrete
  - espone un metodo creazionale parametrico per ottenere factory concrete
- Una factory concreta per ogni tipo di DBMS supportato
  - permette di ottenere oggetti DAO appropriati al corrispondente tipo di DBMS
  - può gestire aspetti quali ottenimento della connessione, autenticazione, ...
- Un oggetto DTO per ogni tipo di entità che si vuole rappresentare
- Una interfaccia DAO per ogni oggetto DTO
- Una implementazione dell'interfaccia DAO di un DTO per ciascun DB supportato

Pattern DAO con classi Factory (2)



DAO Project (1)

Il progetto riprende lo **StudentRepository** della scorsa settimana

- L'oggetto **`it.unibo.paw.dao.StudentDTO` funge da trasporto** tra l'applicazione e la tabella **students** (è un Java Bean al 100%; suffisso DTO aggiunto solo per chiarezza)
- Sono previste **tre versioni degli oggetti DAO**, una per DB2 (completa), una per HSQLDB (appena accennata) e una per MySQL (appena accennata)
  - ai fini di questa semplice applicazione, i tre DBMS accedono ai database nello stesso modo
- **La scelta dell'implementazione da utilizzare** dipende da un particolare parametro specificato in fase di instanziazione del DAOFactory

DAO Project (2)

- Vogliamo accedere alla tabella di nome "xxx" (ad esempio "students" che risiede in un DBMS di tipo "dbmsName" (ad esempio "DB2")

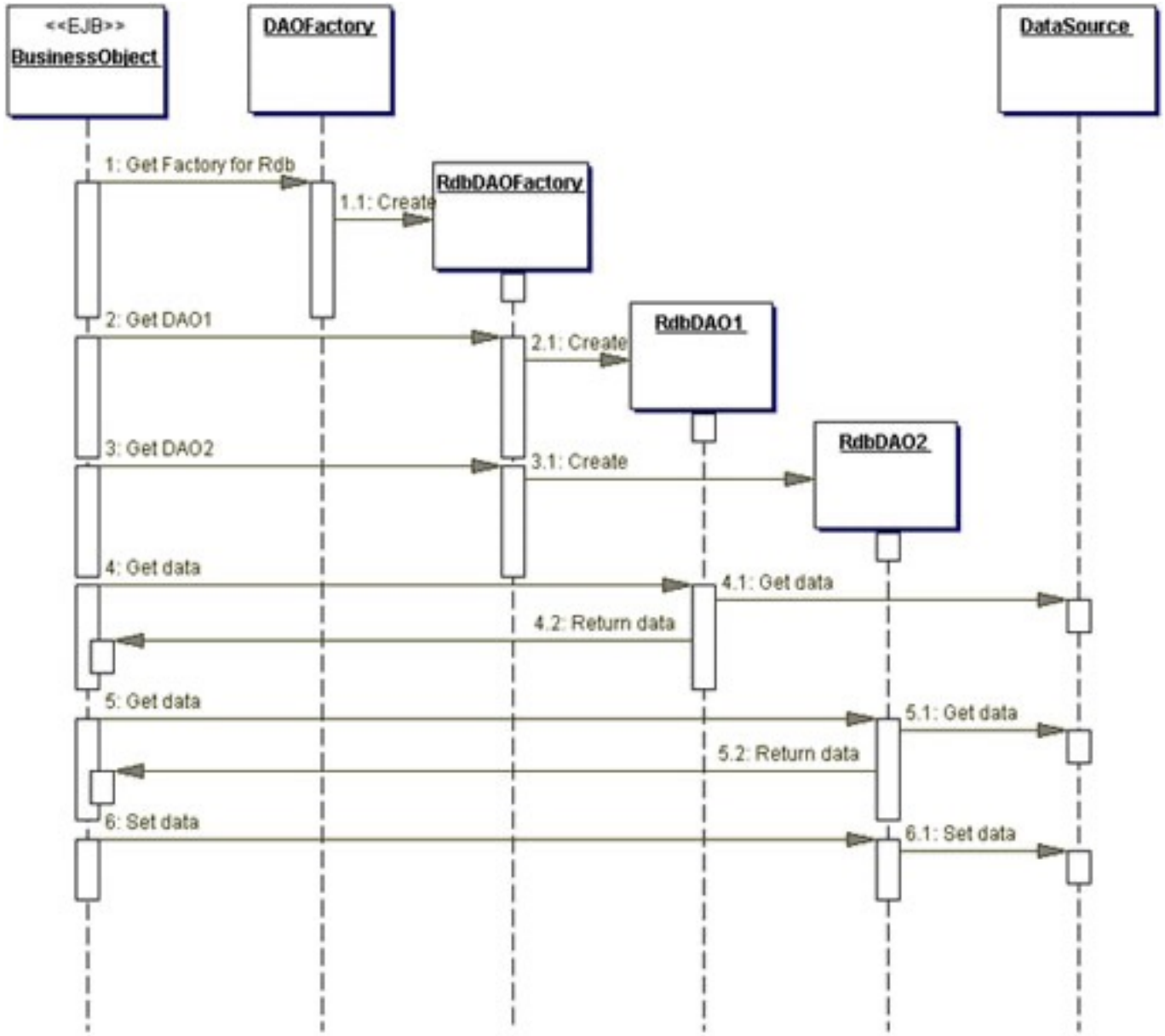
- Un unico **DAOFactory** per applicazione
  - entry-point ai componenti che supportano la persistenza
- Un oggetto Java Bean **xxxDTO** per tabella
  - rappresentazione object-oriented di **una riga** della tabella "xxx"
- Una interfaccia **xxxDAO** per tabella, coi metodi d'accesso alla tabella
  - interfaccia unica per tutti i DBMS, ma **differente tabella per tabella**
- Un dbmsNameDAOFactory per DBMS che si vuole supportare
  - instanziato da DAOFactory, specifica driver JDBC, URI, username, password...
- Per ciascun DBMS, un dbmsNamexxxDAO che implementa l'interfaccia xxxDAO corrispondente
  - implementazione concreta dei metodi dichiarati in xxxDAO per accedere alla tabella "xxx" del DBMS "dbmsName"
  - l'implementazione dei metodi può variare a seconda del dialetto SQL utilizzato dallo specifico DBMS

DAO Project (3)

- Il package **it.unibo.paw.dao** contiene classi e interfacce comuni a tutti i DBMS
- Il package **it.unibo.paw.dao.dbmsName** contiene le classi specifiche necessarie per accedere al DBMS dbmsName

- *All'interno della classe DAOTest, osservate come avviene l'accesso alla tabella **students***
  1) tramite metodo statico DAOFactory si ottiene un riferimento ad una classe **specifica per un particolare DBMS** che estende la classe astratta DAOFactory
  2) tramite tale instanza di DAOFactory si ottiene un riferimento ad una classe che **implementa l'interfaccia StudentDAO**
  3) infine tramite un'instanza della classe che implementa l'interfaccia StudentDAO, si accede alla tabella tramite i metodi offerti dall'oggetto ottenuto

Pattern DAO con classi Factory: diagramma di sequenza



Implementazione degli oggetti DAO

Un consiglio su come procedere nello scrivere il codice degli oggetti DAO basati su JDBC

Ogni metodo

1. dichiari una variabile dove collocare il proprio risultato
2. controlli la bontà dei parametri attuali ricevuti
3. apra la connessione al database
4. formuli gli statement SQL che lo riguardano e imposti il risultato
5. preveda di gestire le eventuali eccezioni
6. rilasci SEMPRE E IN OGNI CASO la connessione in uso
7. restituisca il risultato (eventualmente di fallimento)

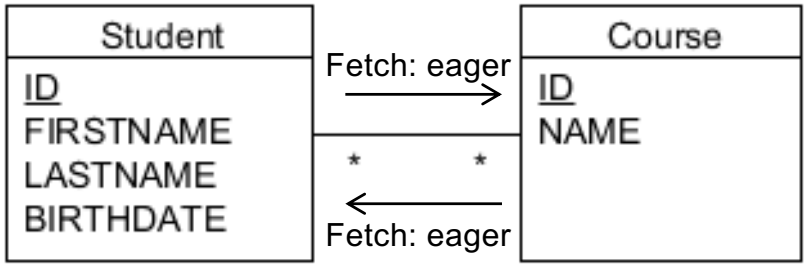
E per quanto riguarda gli **statement SQL** veri e propri

1. crei (se senza parametri) o prepari (se con parametri) lo statement da proporre al database
2. pulisca e imposti i parametri (se ve ne sono, ovviamente)
3. esegua l'azione sul database ed estragga il risultato (se atteso)
4. cicli sul risultato (se presente) per accedere a ogni sua tupla e impostare il proprio risultato con i valori in essa contenuti
5. rilasci la struttura dati del risultato stesso
6. rilasci la struttura dati dello statement

- Potete ritrovare questo schema di operazioni nei commenti a corredo del codice

Ora a voi: DAO (1)

- Partendo dal progetto di esempio «studenti», e considerando il diagramma UML di seguito riportato relativo alla “gestione dei corsi universitari”, si richiede di



- estendere le funzionalità dell'applicazione esistente in modo tale da fornire una soluzione alla gestione della persistenza basata su Pattern DAO
  - in grado di “mappare” efficientemente il modello di dominio rappresentato dai **Java Bean Student**, **Course** con le corrispondenti tabelle derivate dalla progettazione logica applicata al diagramma UML, ovvero **Student**, **Course** e **S-CMapping**, contenute nel DB **TW_STUD**

N.B. La soluzione Java deve mantenere il mapping N-M tra i due Java Bean e rispettare i sensi di percorrenza indicati dalle frecce nell’UML

Ora a voi: DAO (2)

- Nel dettaglio, dopo aver creato gli schemi delle tabelle all'interno del proprio schema nel database **TW_STUD**, implementato i Java Bean e realizzato le classi relative al Pattern DAO per l'accesso CRUD alle tabelle, si richiede la realizzazione di un metodo che permetta
  - dato l'ID di uno studente, di fornire l'elenco dei corsi frequentati
  - dato l'ID di un corso, di fornire l'elenco degli studenti che lo frequentano